CONF-801103--58

A MICROCOMPUTER-BASED EQUIPMENT-CONTROL AND DATA-ACQUISITION SYSTEM FOR FISSION-REACTOR REACTIVITY-WORTH MEASUREMENTS

by

William P. McDowell and R. G. Bucher

Prepared for

IEEE

1980 Nuclear Science Symposium

Orlando, Florida

November 5-7, 1980

UofC-AUA-USDOE

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Operated under Contract W-31-109-Eng-38 for the

U. S. DEPARTMENT OF ENERGY

JHS

A MICROCOMPUTER-BASED EQUIPMENT CONTROL AND DATA ACQUISITION SYSTEM FOR FISSION-REACTOR REACTIVITY-WORTH MEASUREMENTS †

William P. McDowell* and R. G. Bucher**

*Electronics Division
**Applied Physics Division
Argonne National Laboratory
Argonne, Illinois, U.S.A.

ABSTRACT

Material reactivity-worth measurements are one of the major classes of experiments conducted on the Zero Power research reactors (ZPR) at Argonne National Laboratory. These measurements require the monitoring of the position of a servo control element as a sample material is positioned at various locations in a critical reactor configuration. In order to guarantee operational reliability and increase experimental flexibility for these measurements, the obsolete hardware-based control unit has been replaced with a microcomputer based equipment control and data acquisition system. This system is based on an S-100 bus, dual floppy disk computer with custom built cards to interface with the experimental system. To measure reactivity worths, the system accurately positions samples in the reactor core and acquires data on the position of the servo control element. The data are then analyzed to determine statistical adequacy. The paper covers both the hardware and software aspects of the design.

INTRODUCTION

The addition of a small quantity of material into a cavity in a just-critical reactor perturbs the reactivity of the configuration. The material's reactivity-worth is measured by sensing this small change in reactivity associated with the addition of the sample. This change in reactivity is measured by monitoring the change in the mean equilibrium position of a calibrated servo control rod as the desired sample is oscillated into and out of the desired position. A block diagram of the experimental system is shown in Fig. 1. The system consists of the following:

1. A sample drawer drive controlled by a stepping motor to accurately position the desired sample.
2. A sample drawer position transducer to determine sample position to within 0.0005 inch.
3. A drive control to interface processor signals to 1800 oz.-in. stepping motor.
4. The microcomputer system including:
   - 4.1 CPU, RAM, disk controller, serial I/O, and vectored interrupt control.
   - 4.2 Parallel I/O to interface the motor control unit.
   - 4.3 Memory mapped video interface to rapidly display position information.
   - 4.4 Two 24-bit up-down scalers to accumulate data on the position of the servo control element.
   - 4.5 A sample position detector which incorporates direction decoders and a 24-bit up-down scaler.

The sequence of operations for the equipment control and data acquisition system is as follows:

1. Position the reactivity sample at the desired location;

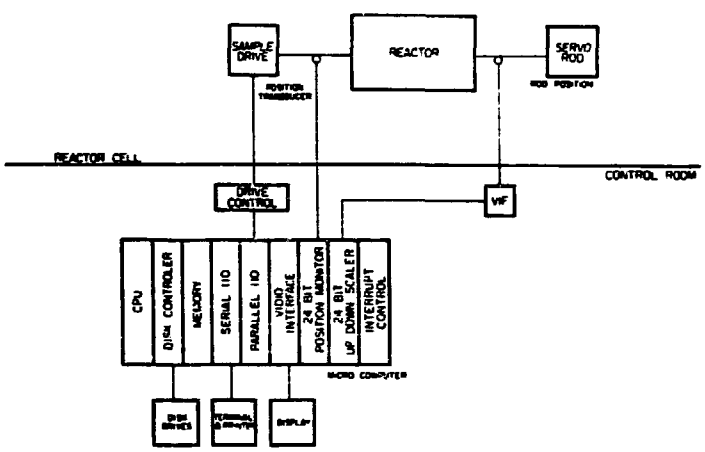

Figure 1. System Block Diagram

† Work performed under the auspices of the U. S. Department of Energy.

2. Execute a preset delay time to allow the reactor to equilibrate;

3. Determine the mean equilibrium position of the servo control element by accumulating data for n intervals of m seconds each.

The sample is then started toward the next position and the process is repeated until the statistics in the position of the control element are satisfactory. The timing of the operation is accomplished with a series of interrupt-driven software counters which record the relative time-of-day, the accumulate time, and the equilibrate time. The interrupt timer is set at the highest operations priority level; the interrupt interval is software selected as 1.00, 0.10, or 0.01 seconds.

Sample positioning is performed by initializing the scaler comparator register with the next position and by driving the sample in the appropriate direction. As the sample passes the desired position, an interrupt is generated which commands the system to back the sample at reduced speed until a second interrupt indicates the desired position has been reached and motion stops.

The applications software for the micro-computer based system is written in 8080A assembler and Fortan IV using Microsoft M80 assembler and F80 compiler. The interrupt service routines are written in assembler, while the experiment operating system and disk I/O operations are written in Fortran.

HARDWARE: EXPERIMENTAL

Samples are positioned at various locations in the reactor assembly through the use of a sample transport mechanism. This device consists of a sample drawer, the sample position mechanism, a stepping motor, a position transducer which is directly connected to the moving structure, and a drive control system which provides drive signals for the stepping motor.

The sample position mechanism consists of the various mechanical assemblies used to support and transport the sample itself. The device has been constructed to accurately position the sample over a ten foot range with an accuracy of 0.0005 inches. The whole of the moving assembly is driven by an 1800 oz.-in. stepping motor which can be driven either in the stepping mode or a high speed mode.

The position transducer is an optical incremental device which generates two square wave signals shifted in phase 90 degrees from each other. The transducer is attached directly to the moving table by means of a rack and pinion gear which eliminates the possibility of slip during the measurement.

The drive control interfaces the computer to the stepping motor. This device consists of a stepping motor translator and a switching system which allows the stepping motor to be driven in the low speed stepping mode or in a high speed synchronous mode. When slewing long distances between the points where samples are to be taken the motor is driven by a two-phase AC signal derived from the power line. When the sample point is reached the motor is switched to the pulse mode of operation and accurately positioned as required by the experiment. All signals from the computer to the drive control are optically isolated to prevent spurious operation of the computer. Solid state relays have been used in the drive control to eliminate arcing.

HARDWARE: MICROCOMPUTER

The microcomputer system is an S-100 based machine which includes 48K bytes of RAM, a disk controller, two floppy disk drives, a vectored interrupt controller, a time and date generator, serial and parallel I/O cards, two 24-bit up-down scalers, one 24-bit up-down scaler with position transducer decoders, and a 1K memory mapped video display controller. The computer used for this experiment is the Intel 8080A and is operated under the CP/M operating system. All the components of the system except the 24-bit scaler boards are commercially available.

A block diagram of the 24-bit scaler board is given in Fig. 2. The interface with the bus is via the address selection logic and the data bus buffers. The board is treated as a sequence of 16 I/O ports located in the I/O space of the 8080A. The address location is selected through the use of DIP switches on the board. The board is designed to be flexible and to allow the experimenter to configure the scalers via software in various ways depending on the particular application. The 24-bit counter can be preset to any desired number as can the 24-bit comparator. The data latch allows the experimenter to take data while the scaler is running without disturbing the count. The comparator can be used to interrupt the processor when the comparator preset is equal to the contents of the scalers. The control register is used to set up the board for the desired operation. The scalers can

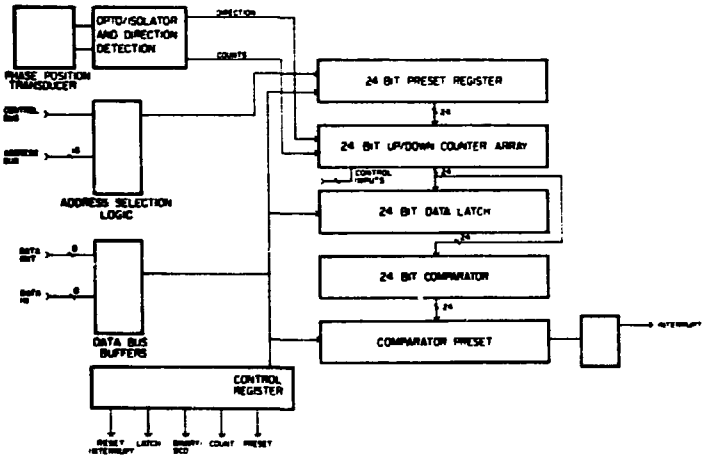

Figure 2. 24-bit Scaler Block Diagram

be set to count in binary or BCD, the count can be enabled, the contents of the preset register can be loaded into the scalers, and the interrupt latch can be preset. On the scaler board when the direction decoder is not being used the scaler can be set to a count up or count down mode.

The incremental encoder generates two quadrature pulse train signals. Directional information is available from the phase relationship between the pulse trains. If phase A lags phase B by 90 degrees then the transducer is moving in the forward direction. If, however, phase A leads phase B by 90 degrees then the transducer is moving in the reverse direction. The number of transitions is a measure of the amount of motion, which in this case is 0.0005 inch/pulse register, and an exclusive-or gate is used to determine the transition points and direction. These signals are then further separated into count signals and up/down information for the 24-bit counter inputs.

SOFTWARE: OPERATING SYSTEM

The experimental operating system, coded as the Fortran IV main program, controls all communications between the user and the experimental system. A set of instructions allows the user to initialize the various experimental and operational parameters and to control the progress of the experiment. These instructions are composed of two, or three, four-character "words," separated by single spaces --- the first is the verb which issues the action (i.e., INIT, READ, STRT); the second is the object which receives the action (i.e., PARM, DRWR, EXPT); and the third, if needed, is a modifier which further specifies the object. For example, the instruction INIT PARM DRWR initializes the experimental parameters controlling the drawer stop positions. If the second instruction "word" is omitted, the default EXPT is assumed. Incorrect or misspelled instructions are flagged by the operating system as "unknown."

The operating system is structured such that there exists three instruction modes:

1. the SET mode in which the user initializes the various system parameters,

2. the RDY (ready) mode in which the user waits to begin execution, and

3. the RUN mode in which the user controls the progress of the experiment.

Only instructions which logically occur within a given instruction mode are accepted by the operating system; otherwise, the instruction is flagged as "illegal."

The instruction HELP produces a listing of all "known" instruction and the corresponding "legal" instruction modes.

Each time the experimental operating system regains control of the operations, the operating system prompts with the message "ENTER INSTRUCTION [XXX MODE] >>," where XXX specifies the current instruction mode. At his leisure, the user inputs the desired instruction. If the instruction is acceptable, that is, neither "unknown" nor "illegal," the operating system transfers control of the operations to the appropriate routine.

The "known" instructions and the corresponding "legal" instruction modes are introduced below. This information is summarized in Table I.

When the operating system is loaded into memory, the instruction mode is initialized to SET. At this point the user may initialize the various experimental and operational system parameters. The instruction INIT DATE sets the date and time on the system's hardware calendar/clock. The instruction INIT DRWR initializes the drawer position scaler and loads the user-input drawer position into the scaler's counters; the instruction READ DRWR reads the current drawer position from the scaler. The INIT DRWR and READ DRWR instructions are legal in any instruction mode, although their execution in the RUN mode is modified slightly. Instructions such as INIT PARM DRWR or INIT PARM TIME are used to modify experimental parameters, such as drawer stop positions and equilibrate and accumulate times, that are stored in a parameters file on the logged-in disk.

When the initialization process has been completed, the instruction INIT EXPT causes the operating system to read the input parameters from the disk file and to print the experimental parameters on the user's console. The system further prompts the user requesting a data file type of 3 alpha-numeric characters (TTT) and a data file description of up to 48 alpha-numeric characters. The operating system creates a data file identification composed of the current date from the hardware calendar/ clock and the input data file type, that is, a data file ID of the form MM/DD/YY.TTT. The logged-in disk directory is searched to determine if the data file already exists. If not, the data file ID is added to the directory and the requested data file is opened; the operating system prints the message "OPEN FILE -- MM/DD/YY.TTT" on the user console. The instruction mode is raised to RDY (ready).

At this point the user has two options --- to start or to quit the experiment. Entering the instruction STRT EXPT writes a data record of type one, initializes the interrupt routines, and enables the timer interrupt. The type one data record contains principally the experimental and operational parameters used for the experiment. The operating system prints the message "ENTER DATA OUTPUT PHASE OF RUN MODE" on the user console. This instruction raises the instruction mode to RUN. However, entering the instruction QUIT EXPT "erases" the data file (deletes the data file ID from the directory) and aborts the experiment. This instruction returns the instruction mode to SET.

When operating in data output phase of RUN mode, the operating system, at the end of each experimental sequence, prints intermediate results on the user's console and writes a data record of type two in the disk data file. Each type two data record contains the accumulated scaler arrays from all pre-selected scalers. To interrupt this output process, the user must press the ESC (escape) key; the operating system prompts with the usual message. At this point the user has four options ---- to stop, to continue, to exit, or to quit the experiment. The instruction STOP EXPT closes the data file and normally terminates the experiment. The operating system prints the message "CLOSE FILE -- MM/DD/YY.TTT" on the user console. The instruction mode is returned to SET. The instruction CONT EXPT is a no-op instruction which continues the output process; the instruction mode remains in RUN. The instruction EXIT EXPT rewinds the data file and allows the experiment to be restarted; the instruction mode is returned to RDY. The instruction QUIT EXPT "erases" the data file and aborts the experiment; the instruction mode is returned to SET. The interrupt system is disabled when the RUN mode is left by any command.

To insure data file security, if either the EXIT or QUIT EXPT instruction is input, the operating system prompts with the message "REWIND or ERASE DATA FILE? YES/--- >>." The user must input YES to execute the instruction; when the operation has been completed the operating system prints the message "REWIND or ERASE FILE -- MM/DD/YY.TTT" on the user console. Otherwise, the operating system ignores the instruction and responds in the RUN mode with the no-op response of the instruction CONT EXPT and in SET or RDY mode with the usual system prompt message.

The instruction QUIT EXPT can be executed from any instruction mode. The operating system always responds by deleting the current data file ID from the logged-in disk's directory and by returning to the SET instruction mode. Consequently, the data file for an experiment normally terminated with the instruction STOP EXPT may be subsequently "erased" with the instruction QUIT EXPT at any point before the next experiment is initiated with an INIT EXPT instruction.

SOFTWARE: DRIVER SYSTEM

When the system is operating in RUN mode, control is shared between the operating system and the driver routine. Normally, the operating system is active, continually sampling the user console for the input of an ESC character and monitoring the status of the system output requests. However, upon the receipt of a timer interrupt, control is transferred to the driver routine which directs the sequencing of the experiment equipment and data accumulation. The execution of this sequencing is also controlled by a set of operational parameters referred to as operation status bytes, OSB's; there exist four OSB's: timer, scaler, drawer, and driver. These driver system parameters are contained in the input parameter file on the logged-in disk. Their initial values can be edited, with caution, in the SET mode with the instruction INIT PARM SYST.

The driver routine is primarily a control module, calling the various timer, scaler, and drawer subroutines at the appropriate interrupts. The discussion of the driver operation is therefore deferred until the operation of each subsystem has been explained.

There is a series of three timers or clocks available to measure the various experimental intervals. Each timer is, in reality, a software counter or scaler which counts the number of interrupts received while the timer is in the active state; a timer is active if the appropriate bit in the timer OSB, TMROSB, is set to one.

Timers #1 and #3 are "one-stage" counters which, after being initialized and activated, are simply incremental with the receipt of each interrupt. Timer #1 is a 32-bit counter; timer #3 is a 16-bit counter. For the present experimental operations, timer #1 measures the relative time-of-day or time-of-run. It is preset to zero and continuously counts the number of interrupts as the experiment progresses; timer #1 is therefore active during all three phases of the experimental sequence. Timer #3 is employed as the equilibrate timer, inserting a preset delay between phases 1 and 3 of the experimental sequence; that is, it is only active during phase 2.

Timer #2 is a "two-stage" counter which, after being initialized and activated, operates as follows: the first stage is incremented at the receipt of each interrupt; after the preset count is obtained, the first stage is reinitialized and the second stage is incremented. Each stage of this timer is a 16-bit counter. Timer #2 is the data accumulate timer, producing n intervals of m interrupts each; the experimental data is collected for each of the n intervals. Timer #2 is active only during phase 3.

The accumulate and equilibrate times are contained in the input parameters file on the logged-in disk; these values can be edited in SET mode with the instruction INIT PARM TIME. The "zero" value for the time-of-run is also contained in this input parameters file. Since under normal operations the input parameter is actually zero, this parameter is included with the driver system parameters and therefore can only be edited with the instruction INIT PARM SYST.

The software counting for each timer is accomplished with the Assembly Language instruction DAD D, where the D, E, register pair is initialized to one. Use of a two-byte arithmetic operation greater increases the maximum scaler count without complicating the program structure with cascaded one-byte operations or affecting significantly execution time for the counting. Use of DAD rp (add register pair) rather than INX rp (increment register pair), although requiring additional execution time, more than compensates in time saved by facilitating the check for the end of the counting interval. By initializing the scaler to the negative of the desired maximum count, the DAD rp instruction will produce a carry flag CY when the counter overflows, that is, when the interval is completed. Interrogating the carry flag is faster than checking even a single byte of the two-byte counter. Timer #3 simply contains a single DAD rp instruction. The two-stage timer #2 incorporates a DAD rp into each stage; the first stage carry flag is checked within the timer #2 routine to determine if the second stage is to be incremented. Timer #1 contains two cascaded DAD rp instructions, the second counting the number of overflows of the first.

Because of the versatility of the scaler design, each must be initialized to operate in the desired mode. This includes specifying the counting mode (up/down and binary/ BCD) and status (enabled/disabled) and designating the latch registers as an input port and the preset and compare registers as output ports. Also the control port for the scaler must be initialized to operate in the bit set/reset mode.

There are two types of hardware data scalers --- the drawer position scaler, designated #1, and the experimental data scalers, designated #2 and #3. Scaler #1 decodes and counts the pulse train from the linear drawer position encoder. The up/down control for this scaler is incorporated in the decoding circuitry and is therefore not included in the scaler initialization. The initialization however does enable the count, which should not be disabled at any time. Furthermore, the scaler's reset circuitry is isolated from the computer's reset bus line, so that re-booting of the computing system does not alter the initialized status of the scaler or the position data in the counter. Consequently, unless it is desired to reset the contents of the scaler, a single INIT DRWR instruction is sufficient each time the computer system is powered-up. The current drawer position entered with the instruction INIT DRWR is converted to the position count, transferred to the preset registers, and initialized onto the counters. Conversely, with the instruction READ DRWR, the contents of the counter are read into latch registers, transferred to memory, and converted to position. The compare function of scaler #1 is used to interrupt the operating system when the contents of the counter equals the preloaded position count in the compare registers.

Scalers #2 and #3 accumulate experimental data, quantities such as flux level or control rod position, from external voltage-to-frequency converters. These scalers are initialized to count up in binary but are disabled. Their operation is controlled principally by timer #2; a simplified description of the operation sequence follows. Immediately prior to the data accumulation phase of the experimental sequence, the scalers are initialized to zero and enabled to count. At the end of each interval of m interrupts all scalers are latched; the data from the latch registers of the selected scalers are transferred to an array in memory. A scaler is selected if the appropriate bit in the scaler OSB, SCLOSB, is set to one. Upon the completion of n intervals, each of these data arrays are written in a record on the disk data file. The experimental data for each given interval is simply the difference between the scaler value for the given interval and that for the previous interval.

The data accumulation procedure is complicated by the need to obtain precise and equal counting intervals of known duration for the hardware data scalers. In order to satisfy these requirements, the following method of scaler operation is employed. During the (i-1)st interrupt service, the scalers are initialized to the value (zero) loaded into the preset register and enabled to count. With the ith interrupt service, the counters are read into latch registers and transferred to arrays in memory; similarly, with the (i+m)th, the (i+2m)th,... and the (i+nm)th interrupt service, the counters are latched and the data are stored. The resulting (n+1) latched data produce n interval values. Since each latch occurs at the same time within the interrupt service, the duration of the intervals are guaranteed to be equal; in addition, the precision in the duration of the counting interval matches that of the interrupts which are generated by the computer's internal clock. Other methods of scaler operation will not efficiently meet these requirements. To implement this method, the timer #2 operation is modified to produce (n+1) counting intervals, where the first interval contains only a single interrupt; the timer #3 interrupt preset is correspondingly decreased by one to preserve the total equilibrate time.

An independent decoding/scaling circuit, designed into the hardware scaler #1 of the microprocessor system, operates in parallel with the external position readout; naturally, all automated operations are controlled by the position information contained in the microprocessor's scaler.

The movement of the drawer oscillator is controlled by two software routines --- a subroutine which is called for initiating drawer motion and an interrupt routine which is serviced upon reaching the stop position. The number of and the values for the desired drawer stop positions are contained in the input parameters file on the logged-in disk; these values can be edited in SET mode with the instruction INIT PARM DRWR. When the experiment is initiated, these values are stored in an array in memory as three-byte integer position counts; a software index records the number of the current stop position.

Movement of the drawer oscillator occurs only during the first phase of the experimental sequence; the details of this positioning operation are discussed below. At the appropriate time in the experimental sequence, the driver routine calls the drawer oscillator subroutine to initiate the drawer motion. This subroutine responds by comparing the current drawer position which is latched from scaler #1 with the desired stop position which is located using the position index. Comparison of these values, a cascaded one-byte comparison of up to three bytes beginning with the most significant byte, determines the direction of drawer travel; the drawer is then set in motion in the computed direction at fast speed. The desired stop position is loaded into the scaler's compare registers and the comparator interrupt is enabled. At this point the subroutine returns to the driver routine. Upon receipt of the first comparator interrupt, indicating that the drawer has reached the desired stop position, the interrupt service routine reverses the direction of travel and decreases the speed to slow. The comparator interrupt is again enabled; the desired stop postion remains in the scaler's compare registers. Upon receipt of the second comparator interrupt, the interrupt service routine stops the drawer motion. This method of positioning causes an initial overshoot of less than a tenth of an inch and a final position within a few thousandths of an inch of the desired stop position.

The driver routine is the control module responsible for the overall coordination of the various subsystems described above. Upon the receipt of each timer interrupt, the driver routine examines the status of the TMROSB and directs the sequence of operations accordingly. The driver first interrogates the TMROSB to determine the status of timer #2 (accumulate timer); if active, the timer #2 routine is called. The driver, in turn, operates on timer #3 (equilibrate timer) and timer #1 (time-of-run). After the execution of either timer #2 or timer #3, the carry flag is examined to determine if that timer has "timed-out," indicating the completion of the corresponding phase of the experimental sequence; if the flag is set, the driver executes the appropriate sequence of instructions to initiate the next phase of the experimental sequence.

Upon completion of timer #2, indicating the end of the accumulate phase, the following set of operations occurs. First the drawer position and cycle indices are advanced --- the position index contains the number of completed passes through the set of stop position; subsequently, the drawer subroutine is called to initiate drawer motion. Timer #3 is initialized to its preset values and the status of timer #2 is reset to inactive. Finally, the driver OSB, DRVOSB, is set to flag to operating system indicating that the accumulated data is ready for output to the data file on the logged-in disk.

Upon the completion of timer #3, indicating the end of the equilibrate phase, the following set of operations occurs. Timer #2 is initialized to its modified preset values (see section on scaler operation) and the TMROSB is to deactivate timer #3 and activate timer #2. Finally, the current value for the following quantities is saved: the position and cycle indices, the timer #1 value (time-of-run), and the scaler #1 value (drawer position).

TABLE I. Operating System Instructions

Instruction[a]			Mode[b]			
Verb	Object	Modifier	SET	RDY	RUN[c]	Operation (Comments)
INIT	DATE		X	-	-	to initialize date and time
INIT	PARM		X	-	-	to initialize experimental parameters TIME, DRWR, and FARC
INIT	PARM	[TIME]	X	-	-	to initialize equilibrate and accumulate timers
INIT	PARM	[DRWR]	X	-	-	to initialize drawer stop positions
INIT	PARM	[FARC]	X	-	-	to initialize FAR calibration data
INIT	PARM	SYST	X	-	-	to initialize operating system parameters
INIT	DRWR		X	X	X[d]	to initialize drawer position
READ	DRWR		X	X	X[d]	to read/display drawer position
INIT	[EXPT]		X →	-	-	to initialize experiment (to open disk file)
STRT	[EXPT]		-	X →	-	to start experiment (to write in disk file, type one data record)
STOP	[EXPT]		- ←	-	X	to terminate experiment (to close disk file)
CONT	[EXPT]		-	-	X[e] ↻	to continue experiment (to write in disk file, type two data records)
EXIT	[EXPT]		-	- ←	X	to ready system to restart experiment (to rewind disk file)
QUIT	[EXPT]		X ↻	X ←	X ←	to abort experiment (to delete disk file)

[a] [ ] indicates optional instruction "word".

[b] Arrows indicate changes in instruction mode.

[c] Press ESC key to enter instruction in RUN mode.

[d] Automatic return to RUN mode after instruction is executed.

[e] No-operation, return to data output phase of RUN mode.